Kubernetes Services

A diagram of a service

AI-generated content may be incorrect.

**🔹 What is a Kubernetes Service?**

In Kubernetes, Pods are **immutable**—they can die and get recreated with a new IP address. A **Service** provides a stable network endpoint (a DNS name and IP address) to a **group of Pods**. This grouping is usually defined by **labels and selectors**.

**🔹 Why Do We Need Services?**

1. **Pods are temporary**: They may restart or be recreated, getting a new IP each time.
2. **Internal communication**: One component (say frontend) needs to talk to another (say backend).
3. **Load balancing**: Services route traffic to multiple pod instances (replicas).
4. **Exposing apps**: You might want to expose internal apps within the cluster or to the outside world.

A screenshot of a computer

AI-generated content may be incorrect.

PODs communicate with each other through internal networking.

Let’s start with external communication.

Q.) We deployed our POD having a web application running on it. How do we as an external user access the web page?

* The Kubernetes Node has an IP address say 192.168.1.2.
* My laptop is on the same network as well, so it has an IP address 192.168.1.10.
* The internal POD network is in the range 10.244.0.0 and the POD has an IP 10.244.0.2.
* Clearly, I cannot ping or access the POD at address 10.244.0.2 as it’s in a separate network.

So, what are the options to see the webpage?

* First, if we were to SSH into the Kubernetes node at 192.168.1.2, from the node, we would be able to access the POD’s webpage by doing a curl or if the node has a GUI, we could fire up a browser and see the webpage in a browser following the address http://10.244.0.2.
* But this is from inside the Kubernetes Node and that’s not what I really want. I want to be able to access the web server from my own laptop without having to SSH into the node and simply by accessing the IP of the Kubernetes node. So, Kubernetes we need something in the middle to help us map requests to the node from our laptop through the node to the POD running the web container.
* That is where the Kubernetes service comes into play.
* One of its use cases is to listen to a port on the Node and forward requests on that port to a port on the POD running the web application. This type of service is known as a **NodePort** service.

A screenshot of a computer

AI-generated content may be incorrect.

🔹 Types of Kubernetes Services

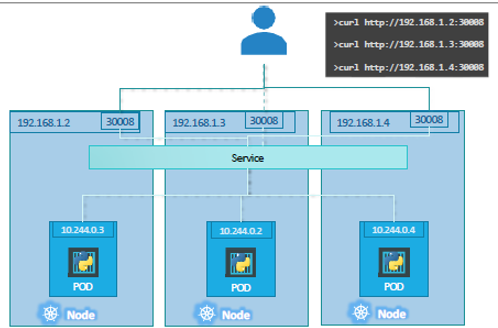
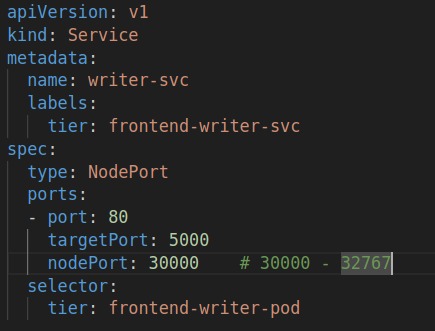
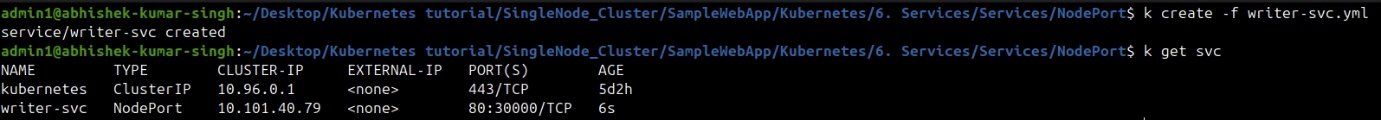
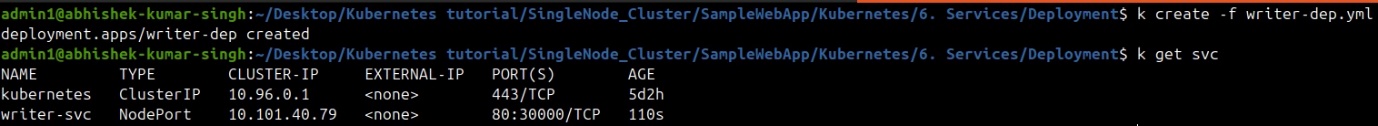
* **NodePort**   
    
  A screenshot of a computer

  AI-generated content may be incorrect.
  + The service makes an internal POD accessible on a Port on the Node.
  + **port range**: 30000–32767 by default.
  + 3 ports involved.
    - Port on the POD 🡪 TargetPort
    - Port on the service 🡪 Port
    - Port on the Node 🡪 NodePort (which we use to access externally)  
        
      A screenshot of a computer

      AI-generated content may be incorrect.
  + 💡 Access from outside: http://<NodeIP>:30008
  + **Note:** **Port** is mandatory field.
    - If we don’t provide a **TargetPort** it is assumed to be the same as Port.
    - And if we don’t provide a **NodePort** a free port in the valid range between 30000 and 32767 is automatically allocated.
  + Inside the cluster Service has its own IP address, called the Cluster-IP of the service.
  + You can have multiple such port mappings within a single service.
  + Use labels and selectors to link PODs
  + Run: kubectl create -f <service-definition>.yml
  + List of services: kubectl get svc

A screenshot of a computer

AI-generated content may be incorrect.

* + Service automatically selects all the PODs to forward the external requests coming from the user.
  + Uses **Random algorithm** to balance load.
  + Thus, the service acts as a built-in load balancer to distribute load across different PODs.  
    
  + What happens when the PODs are distributed across multiple nodes?
    - When we create a service, without us having to do any kind of additional configuration, Kubernetes creates a service that spans across all the nodes in the cluster and maps the target port to the Same NodePort on all the nodes in the cluster.
    - This way you can access your application using the IP of any node in the cluster and using the same port number.
  + To summarize – in ANY case whether it be a single pod in a single node, multiple pods on a single node, multiple pods on multiple nodes, the service is created the same without you having to do any additional steps during the service creation.
  + When PODs are removed or added the service is automatically updated making it highly flexible and adaptive.
  + Examples:  
      
      
      
      
      
      
      
      
      
    Run: <http://localhost:30000> or http://192.168.0.110:30000

* **ClusterIP** (**default**)  
  + creates a **virtual IP address** (called **Cluster IP**) inside the **Kubernetes cluster network** that **routes traffic to a group of backend Pods**.
  + **Scope**: Accessible **only within the cluster**
  + **Use case**: For internal communication between Pods/services like frontend ↔ backend, or app ↔ database

A computer screen with text and numbers

AI-generated content may be incorrect.

* **targetPort**: Actual port on the container that receives the traffic
* When this service is created:  
  + A **ClusterIP** is allocated (e.g., 10.96.0.10)
  + A **DNS entry** is created: **db-svc**.default.svc.cluster.local
  + **kube-proxy** on each node sets up **iptables** or **IPVS** rules to forward traffic to the selected Pods

| **Access Location** | **Access Method** | **Works?** |
| --- | --- | --- |
| Inside any Pod | Use DNS name or ClusterIP | ✅ |
| From outside cluster | Not directly accessible | ❌ |

* **LoadBalancer**
  + Access: External (creates a cloud provider's external load balancer).
  + **Use case**: For production-grade exposure.
  + **Requires**: Cloud provider support (AWS, GCP, Azure).  
    A screen shot of a computer

    AI-generated content may be incorrect.
  + 💡 A public IP is provisioned by the cloud provider.
* **ExternalName**
* **Access**: Maps a service to an **external DNS name**.
* **Use case**: Accessing external services like a database hosted outside Kubernetes.  
  A black background with green text

  AI-generated content may be incorrect.
* 💡 No selector or pod targeting; it works purely at the DNS level.

**🔹 Summary Table**

| **Service Type** | **Internal** | **External** | **Load Balancer** | **Use Case** |
| --- | --- | --- | --- | --- |
| ClusterIP | ✅ | ❌ | ❌ | Internal communication |
| NodePort | ✅ | ✅ | ❌ | Basic external access |
| LoadBalancer | ✅ | ✅ | ✅ | Cloud apps needing public access |
| ExternalName | ❌ | ✅ | ❌ | Connect to external services |